National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Interagency Report 7696
(DRAFT)

Common Platform Enumeration : Name Matching Specification Version 2.3 (DRAFT)

Mary C. Parmelee
Harold Booth
David Waltermire

DRAFT

7

8

NIST Interagency Report 7696 (DRAFT)

Common Platform Enumeration: Matching Specification Version 2.3 (DRAFT)

Mary C. Parmelee
Harold Booth
David Waltermire

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

August 2010

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, Director

DRAFT

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report 7696 (DRAFT)
30 pages (August 2010)

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

DRAFT

Acknowledgments

The authors, Mary C. Parmelee of the MITRE Corporation, and Harold Booth and David Waltermire of NIST wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge Paul Cichonski of Booz Allen Hamilton, Seth Hanford of Cisco Systems, Inc., Tim Keanini of nCircle, Kent Landfield of McAfee, Inc., Brant A. Cheikes of the MITRE Corporation, Jim Ronayne of Cobham plc, and Shane Shaffer of G2, Inc. for their insights and support throughout the development of the document.

DRAFT

Abstract

Following security best practices is essential to maintaining the security and integrity of today's Information Technology (IT) systems and the data they store. Given the speed with which attackers discover and exploit new vulnerabilities, best practices need to be continuously refined and updated at least as fast as the attackers can operate. To meet this challenge, *security automation* has emerged as an advanced computer-security technology intended to help information system administrators assess, manage, maintain and upgrade the security posture of their IT infrastructures regardless of their enterprises' scale, organization and structure. The United States government, under the auspices of the National Institute of Standards and Technology (NIST), has established the Security Content Automation Protocol (SCAP—cf. scap.nist.gov) to foster the development and adoption of security automation specifications and data resources.[1]

The foundation of an effective security automation system is the capability to completely and unambiguously characterize the software systems, hardware devices and network connections which comprise an enterprise's computing infrastructure. With a detailed computing asset inventory in hand, one can begin to integrate and correlate a wealth of other knowledge about, e.g., vulnerabilities and exposures,[2] configuration issues and best-practice configurations,[3] security checklists,[4] impact metrics,[5] and more. In order for heterogeneous security automation systems to effectively share asset inventory information they must adopt common non-proprietary methods that enable its seamless exchange throughout the security information and event management lifecycle.

The *Common Platform Enumeration* (CPE) is a family of specifications that are aimed at addressing the security automation community's need for a standardized method to identify and describe the software systems and hardware devices present in an enterprise's computing asset inventory. Collectively, the CPE specification stack aims to deliver these capabilities to the security automation community:

- A method for assigning unique machine-readable identifiers to certain classes of IT products and computing platforms;
- A method for curating (compiling and maintaining) dictionaries (repositories) of machine-readable product and platform identifiers;
- A method for constructing machine-readable referring expressions which can be mechanically compared (i.e., by a computer algorithm or procedure) to product/platform identifiers to determine whether the identifiers satisfy the expressions;
- A set of interoperability requirements which guarantee that heterogeneous security automation tools can select and use the same unique identifiers to refer to the associated products and platforms.

DRAFT

[1] For more information on SCAP, cf. NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol*, http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf.
[2] See, e.g., MITRE's Common Vulnerabilities and Exposures (CVE) project, on the web at cve.mitre.org.
[3] See, e.g., MITRE's Common Configuration Enumeration (CCE) project, on the web at cce.mitre.org, and also the Federal Desktop Core Configuration (FDCC), on the web at fdcc.nist.gov.
[4] See, e.g., the National Checklist Program Repository, on the web at checklists.nist.gov.
[5] See, e.g., the Common Vulnerability Scoring System, on the web at nvd.nist.gov/cvss.cfm.

Audience

This specification document defines standardized methods for matching CPE names. These methods are envisaged to be of interest to:

1. **Asset inventory tool developers.** Asset inventory tools inspect computing devices and assemble catalogs listing installed component hardware and software elements. In the absence of CPE, there is no mechanism for how these tools should report what they find. The CPE Specification Stack provides all the technical elements needed to comprise such a capability. Furthermore, CPE is intended to address the needs of asset inventory tool developers regardless of whether the tools have credentialed (authenticated) access to the computing devices subject to inventory.

2. **Security content automation tool developers.** Many security content automation tools are fundamentally concerned with making fully- or partially-automated information system security decisions based on collected information about installed products. The CPE Specification Stack provides a framework that supports correlation of information about identical products installed across the enterprise, and association of vulnerability, configuration, remediation and other security-policy information with information about installed products.

3. **Security content authors.** Security content authors are concerned with creating machine-interpretable documents that define organizational policies and procedures pertaining to information systems security, management and enforcement. Often there is a need to tag guidance, policy, etc., documents with information about the product(s) to which the guidance, policy, etc., applies. These tags are called *applicability statements*. The CPE Specification Stack provides a standardized mechanism for creating applicability statements which can be used to ensure that guidance is invoked as needed when the product(s) to which it applies is discovered to be installed within an enterprise.

DRAFT

Table of Contents

DRAFT

List of Figures and Tables

DRAFT

1. Introduction

Following security best practices is essential to maintaining the security and integrity of today's Information Technology (IT) systems and the data they store. Given the speed with which attackers discover and exploit new vulnerabilities, best practices need to be continuously refined and updated at least as fast as the attackers can operate. To meet this challenge, *security automation* has emerged as an advanced computer-security technology intended to help information system administrators assess, manage, maintain and upgrade the security posture of their IT infrastructures regardless of their enterprises' scale, organization and structure. The United States government, under the auspices of the National Institute of Standards and Technology (NIST), has established the Security Content Automation Protocol (SCAP—cf. scap.nist.gov) to foster the development and adoption of security automation specifications and data resources.[6]

The foundation of an effective security automation system is the capability to completely and unambiguously characterize the software systems, hardware devices and network connections which comprise an enterprise's computing infrastructure. With a detailed computing asset inventory in hand, one can begin to integrate and correlate a wealth of other knowledge about, e.g., vulnerabilities and exposures,[7] configuration issues and best-practice configurations,[8] security checklists,[9] impact metrics,[10] and more. In order for heterogeneous security automation systems to effectively share asset inventory information they must adopt common non-proprietary methods that enable its seamless exchange throughout the security information and event management lifecycle.

1.1 Purpose

The Common Platform Enumeration (CPE) addresses the security automation community's need for a standardized method to identify and describe the software systems and hardware devices present in an enterprise's computing asset inventory. Four specification documents comprise the CPE stack:

1. The *Naming* specification defines the logical structure of well-formed CPE names (WFNs) and the procedures for binding and unbinding WFNs to their encodings to and from machine-readable encodings;
2. This document, the *Matching* specification defines the procedures for comparing source to target CPE names to determine whether they refer to some or all of the same products or platforms.
3. The *Dictionary* specification defines the concept of a dictionary of names, and prescribes high-level rules for dictionary creators.
4. The *Language* specification defines a standardized structure for forming complex logical expressions from WFNs.

These four specifications are arranged in a *specification stack* as depicted in Figure 1-1.

[6] For more information on SCAP, cf. NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol*, http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf.
[7] See, e.g., MITRE's Common Vulnerabilities and Exposures (CVE) project, on the web at cve.mitre.org.
[8] See, e.g., MITRE's Common Configuration Enumeration (CCE) project, on the web at cce.mitre.org, and also the Federal Desktop Core Configuration (FDCC), on the web at fdcc.nist.gov.
[9] See, e.g., the National Checklist Program Repository, on the web at checklists.nist.gov.
[10] See, e.g., the Common Vulnerability Scoring System, on the web at nvd.nist.gov/cvss.cfm.

DRAFT

Figure 1-1: CPE Specification Stack

Collectively, the CPE Specification Stack aims to deliver these capabilities to the security automation community:

- A method for assigning unique machine-readable identifiers to certain kinds of IT products and platforms;
- A method for compiling and maintaining dictionaries (repositories) of machine-readable product and platform identifiers;
- A method for constructing machine-readable referring expressions which can be mechanically resolved (i.e., by a computer algorithm or procedure) against one or more dictionaries to yield sets of candidate referents;
- A set of interoperability requirements which guarantee that heterogeneous security automation tools can select and use the same unique identifiers to refer to the associated products and platforms.

The primary purpose of the CPE Name Matching specification is to provide a method for a one-to-one comparison of two CPE Names according to the matching methods specified in Sections 5, 6 and 7 of this document.

1.2 Name Matching Scope

This section specifies what functional considerations are in and out of scope for this specification.

1.2.1 In Scope

Functionally, the scope of CPE Name matching includes a one-to-one comparison of a source CPE name to a target CPE Name. The matching capability described in this specification encompasses two main parts: an attribute comparison method for individual attribute values within a CPE name and a minimal common name matching method. Taken together, these two parts provide for basic tool interoperability, while remaining flexible and extensible enough to apply to the broadest range of use cases, including unanticipated use cases.

CPE Name matching returns individual outcome results for attribute comparison along with a single overall result for a name match. Name matches are intentionally minimally defined and extensible. Name matches are defined in terms of a given set relationship between the source name and target name.

CPE Name matching as described in this specification has limited scope when applied to a list of CPE Names. Specifically, CPE name matching can sequentially compare a single source name to a list of target names until such time that the first name match is found in the list.

DRAFT

1.2.2 Out of Scope

The name matching method that is specified in this document may be used as the foundation for defining more complex matching capabilities at higher levels of the CPE specification stack. The following aspects of CPE name matching are outside the scope of the current CPE Name Matching specification:

1. **Multiple name results.** Although CPE Name matching can be sequentially applied to a list of target names, it returns only the first match found in the list. Returning lists of results is out of scope.
2. **Many-to-many list comparisons.** When comparing a list of source names to a list of target names, the CPE Name Matching specification provides a foundation from which to build list-to-list comparisons, but specifying many-to-many comparisons is currently out of scope.
3. **Weighting of matching results**. Although CPE Name matching provides results for partial or possible matches, determining whether or not one match is more relevant than another is out of scope. For example, the algorithm does not distinguish whether a match of a version attribute value is more or less relevant than a match of a language attribute value.
4. **CPE Language matching** is out of scope. It will be specified in the CPE Language Specification.

1.3 Normative References

The following documents are indispensible references for understanding the application of this specification.

[CPE22] Buttner, A. and N. Ziring. (2009). *Common Platform Enumeration—Specification*. Version 2.2 dated 11 March 2009. See: http://cpe.mitre.org/specification/spec_archive.html.

[CPE23-D] Cichonski, P. and Waltermire, D. (2010). *Common Platform Enumeration: Dictionary.*

[CPE23-L] Waltermire, D. and Cichonski, P. (2010). *Common Platform Enumeration: Language.* Version 2.3.

[CPE23-N] Cheikes, B. A. and Waltermire, D. (2010). *Common Platform Enumeration: Naming.*

[RFC2119] Bradner, S. (1997). *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. See http://www.ietf.org/rfc/rfc2119.txt.

1.4 Document Structure

This specification document is organized as follows:

- Section 1 provides an introduction and overview of security automation, the purpose for the CPE specification, and the purpose for and scope of the CPE Name Matching specification. It also provides information about this document's structure and normative references;
- Section 2 defines the key terms and abbreviations used in this specification;
- Section 3 defines what it means for a software product to conform with this specification;
- Section 4 places this specification in the context of related specifications and standards;
- Section 5 describes the foundational concepts, constructs and notations associated with this specification;

DRAFT

- Section 6 describes CPE Name matching methods;
- Section 7 describes expected name matching behavior in pseudo-code.
- Appendix A documents per-release changes to this specification over time.

1.5 Document Conventions

Relevant conventions that are applied to the content of this specification include assigning special meaning to text based on Font usage, restricted usage of requirements related terminology, and notation conventions for reference citation.

1.5.1 Font Usage

Text intended to represent computing system input, output, or algorithmic processing is presented in `fixed-width Courier font`.

1.5.2 Terminology Usage

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

1.5.3 References

Normative references are listed in Section 1.3 of this document. The following reference citation conventions are used in the text of this document:

- For normative references, a square bracket notation containing an abbreviation of the overall reference citation, followed by a colon and subsection citation where applicable (e.g. [CPE-N:5.2.1] is a citation for CPE Naming specification, Section 5.2.1);
- For references within this document (internal references) and non-normative references, a parenthetical notation containing the “cf.” (compare) abbreviation followed by a section number for internal references or an external reference, (e.g. (cf. 2.1.4) is a citation for Section 2.1.4of this document).

DRAFT

2. Terms, Definitions and Abbreviations

2.1 Terms and Definitions

The following terms and definitions apply to the CPE Name Matching specification. Where practical, we have adapted terms and definitions from authoritative sources, such as industry, national and international standard specifications. These sources are cited as appropriate.

2.1.1 Attribute

In the context of the CPE Version 2.3 family of specifications, an attribute is a property or characteristic of a computing product. In CPE 2.2 the term "component" was commonly used for this purpose. We have adopted the new term "attribute" in CPE 2.3 in order to clarify the distinction between CPE 2.2 'components" and computing components, such as software modules. Examples of CPE 2.3 attributes are: part, vendor, product, and version. CPE attributes and their value constraints are defined in the CPE Naming specification [CPE-N:5.4, 5.5].

2.1.2 Attribute-Value Pair

An *attribute-value pair* is a tuple *a=v* in which *a* (the *attribute*) is an alphanumeric label representing a property or state, and *v* (the *value*) is the value assigned to the attribute.

2.1.3 Bind

In general terms, to *bind* means to connect two things together. In the context of this specification, to *bind* means to deterministically transform a logical construct into a machine-readable representation suitable for machine interchange and processing. The result of this transformation is called a *binding*. A binding may also be referred to as the "bound form" of its associated logical construct.

2.1.4 CPE Attribute Comparison

The first phase of CPE name matching where a matching engine compares each of the 11 *attribute-value pairs* of a source CPE name to the corresponding *attribute-value pairs* of a target name according to the matching method specified in Sections 5 and 6 of this document. CPE name matching is based on the set of combined outcomes of a CPE attribute comparison, which specifies one of the six possible logical attribute comparison outcomes for each attribute in a CPE name.

2.1.5 CPE Name Match

A *CPE name match* occurs when a source CPE name is found by a matching engine to have a defined relationship to a target CPE name. A CPE name match is determined by the combined outcome results of its constituent attribute comparison (cf. 2.1.4) as specified in Sections 5 and 6 of this document.

2.1.6 Escape

For the purposes of CPE, the term *escape* means to precede non-alphanumeric characters (e.g. *, $, ?) with the backslash (\ ) escape character in a value string. When a non-alphanumeric character is escaped in a Well Formed CPE name, it shall be processed as string data. When a non-alphanumeric character is

DRAFT

un-escaped in a well formed CPE name, it shall be interpreted as a special character as specified in at least one CPE 2.3 specification.

2.1.7 Matching

The CPE Name Matching specification defines two phases of matching: attribute comparison (cf. 2.1.4) and name matching (cf 2.1.5). Matching is always a one-to-one source-to-target comparison of CPE values. CPE name matching compares source-to-target attribute values at the attribute comparison level, and then applies rules to the set of attribute outcomes to determine a name match. A detailed technical description of Matching is provided in Sections 5 and 6 of this document.

2.1.8 Product

In the context of CPE *product* refers to a computing product consisting of one of the following three types:

1. Application;
2. Operating system;
3. Hardware device.

The terms application, operating system, and hardware device are defined in the CPE Naming specification [CPE-N2.1.1, 2.1.9, 2.1.8].

2.1.9 Source Name

In the context of CPE name matching, a source name is a single well-formed CPE name (WFN) that a matching engine compares to a target CPE well-formed name to determine whether or not there is a source-to-target match. In CPE 2.2 terms this is the X value in the CPE 2.2 matching algorithm.

2.1.10 Special Character

A special character is a non-alphanumeric character that is defined by one or more CPE specifications to have a special meaning when it appears un-escaped in a WFN. Special characters typically trigger a processor to perform a given function. The rules for escaping CPE special characters are specified in the CPE Naming specification [CPE-N].

2.1.11 Target Name

In the context of CPE name matching, a target name is a single well-formed CPE name that is the target of a matching process. A matching engine compares a source CPE name to a target CPE name in order to determine whether or not there is a source-to-target match. In CPE 2.2 terms a target name is a single item in the list of known values (each N of K) and is the N value in the CPE 2.2 Matching algorithm.

2.1.12 Value String

A *value string* assigned to an attribute of a WFN must be a *non-empty contiguous string of bytes* encoded using the American Standard Code for Information Interchange (US-ASCII, also known as ANSI_X3.4-1968).

DRAFT

2.1.13 Well-Formed CPE Name

A *well-formed CPE name* (WFN) is defined to be a logical construct that constitutes an unordered list of 11 attribute-value pairs that describe or identify a software application, operating system, or hardware device. By *unordered* we mean that there is no prescribed order in which attribute-value pairs should be listed, and there is no specified relationship (hierarchical, set-theoretic or otherwise) among attributes. WFNs must satisfy the criteria specified in the CPE Naming specification [CPE-N:5.2.1]. For a full description and basic usage constraints on WFN logical attribute values, see Section 5 of the CPE Naming specification [CPE23-N:5].

2.2 Abbreviated Terms

The following abbreviations and acronyms are used in this specification.

ANSI	American National Standards Institute
CPE	Common Platform Enumeration
DLL	Dynamic Link Library
FDCC	Federal Desktop Core Configuration
GNU	GNU's Not Unix (recursive acronym)
IT	Information Technology
NIST	National Institute of Standards and Technology
SCAP	Security Content Automation Protocol
US-ASCII	United States - American Standard Code for Information Interchange
WFN	Well-formed name

DRAFT

3. Conformance

A product manufacturer or product vendor can claim product conformance with the CPE Name Matching specification when it implements the behavior that is specified in the pseudo-code and produces the identical results for CPE attribute comparison outcomes that are specified in this document (cf. 5, 6, 7).

DRAFT

4. Relationship to Existing Specifications and Standards

4.1 CPE Specification Version 2.2 and ISO 19770-2

The CPE Naming specification describes the relationship between the CPE 2.3 family of specifications, the CPE Specification Version 2.2, and ISO 19770-2 [CPE-N:4.1, 4.3]

4.2 CPE Naming Specification

The CPE Name Matching specification builds on the foundation of the CPE Naming specification [CPE23-N]. Many of the concepts and methods that are applied in this specification are defined in the CPE Naming specification. For this reason, we strongly recommended that CPE consumers read the CPE Naming specification before reading the Name Matching specification.

4.3 CPE Dictionary Specification

For the purposes of forming CPE dictionary names, the CPE Dictionary specification restricts the usage of the special characters question mark (?) and asterisk (*) as they are interpreted in this specification. The CPE Dictionary also builds upon the CPE Name Matching algorithm to define additional matching functionality specific to CPE Dictionary maintenance and use.

4.4 CPE Language Specification

The CPE Language Matching section of the CPE Language specification is built on the foundation of the WFN matching concepts and methods that are defined in this specification.

DRAFT

5. Name Matching Overview

CPE Name matching specifies a common set of capabilities for matching sets of unordered attribute-value pairs known as CPE Well Formed Names (WFNs). It specifies a method for a one-to-one comparison of a source WFN to a target WFN. CPE Name Matching specifies two phases of matching: attribute comparison and name matching. It aims to specify common matching functionality in order to facilitate interoperability, while remaining flexible enough to apply to the broadest range of use cases, including unanticipated use cases. To this end, the CPE Name Matching specification defines a common comparison method at the attribute level and a minimal common matching method at the name level.

Decisions about what constitutes a CPE Name match are precision vs. recall design trade-offs that are typically use case dependent. For example, in the case where the source WFN is generated from the sparse results of a non-authenticated asset inventory tool, it is reasonable to decide that only a small number of matching attributes constitutes a CPE Name match. In contrast, in cases where both the source and target WFNs are fully specified CPE Dictionary names, it is reasonable to decide that a name-level match occurs only when there is an exact match of all attribute values. For this reason, we intentionally leave the majority of decisions about what constitutes a CPE Name match to be decided by CPE implementers at design time. This section describes the foundational concepts, constructs and notations associated with CPE Name matching.

5.1 Name Matching Concepts

In order to understand CPE Name matching, it is essential to understand the meaning of the concepts described in this section. The descriptions here build upon the term definitions in Section 2.1of this document as well as the concepts described in the CPE Naming specification [CPE-N]. In order to understand the context of the descriptions in this section, CPE consumers should first read the CPE Naming specification and then Section 2.1 of this document.

5.1.1 Well Formed Names

CPE Name matching is defined independently of any bound form of a CPE Name. Rather, it is defined only in terms of the logical constructs of a WFN; namely the attribute and special character values described in Section 5.1.2 and 5.1.3 of this document. In their bound form, these values are typically associated with character values such as the dollar sign ($), hyphen (-), and asterisk (*); or embedded special characters, such as the asterisk (*) and question mark (?).

5.1.2 Logical CPE Attribute Values

The following three logical CPE attribute values for WFNs are defined in the CPE Naming specification:

1. ANY – Any value is acceptable. The value does not matter;
2. NA – the value is not applicable. No value exists for the attribute;

For a full description and basic usage constraints on WFN logical attribute values, see the CPE Naming specification [CPE23-N:5].

5.1.3 Special Characters

The CPE Naming specification designates two *special characters* for use in the CPE attribute value strings of a WFN. When these characters appear un-escaped within a CPE attribute value string, they

DRAFT

may be interpreted as having a special meaning by CPE specifications that are higher in the CPE stack [CPE-N:5.5.2].

1. Asterisk (*)
2. Question mark (?)

This specification (CPE Name Matching) assigns special interpretations to the asterisk and question mark special characters. An un-escaped asterisk that is embedded within a CPE attribute value string is interpreted as a multi-character wild card. An un-escaped question mark that is embedded within a CPE attribute value string is interpreted as a single character wild card. Logically, these wild cards translate to multi-character ANY and single character ANY respectively.

5.2 Un-escaped Character Matrix

This specification does not require that all security automation tools transform CPE Names into WFNs prior to matching. In practice, security automation tool developers may choose to unbind CPE Names to WFNs prior to matching as we have in this specification, or they may apply matching to the bound form of their choice. However, for the purpose of security automation tool interoperability, we specify the following requirements:

1. The logical meaning of un-escaped CPE characters, including unspecified characters, must be applied as defined in the CPE Naming specification [CPE-N]
2. The logical meaning that is applied to the embedded characters asterisk (*) and question mark (?) must be applied as specified in this specification.

Table 5-1 below summarizes the relationship between the set of un-escaped characters that are relevant to the CPE Name Matching specification. Column 1, “Un-escaped Characters” describes each un-escaped character including the absent or unspecified character. Column 2, “CPE Name Form(s)” lists the forms of CPE names in which the un-escaped characters can legally appear according to the CPE Naming specification [CPE-N]. Column 3, “Logical Meaning”, lists the logical meaning that each un-escaped character designates. This logical meaning is consistent across all CPE 2.3 specifications. Note that the “Logical Meaning” column describes the semantics of un-escaped characters in terms of CPE Logical Values. However, this description is not to be confused with CPE Logical Attribute Value notation in WFNs. CPE Logical Value notation (ANY and NA) designates only whole attribute values as described in Section 5.1.2 of this specification. The CPE notation for the logical meaning ANY within the value string of a CPE attribute is designated by a question mark (?) for a single character and an asterisk (*) for multiple characters. For more detail on CPE notation see the CPE Naming specification [CPE-N].

Table 5-1: Un-escaped Character Matrix

Un-escaped Characters	CPE Name Form	Logical Meaning
Hyphen (-)	URI/Formatted string	NA
Blank ( )	URI	ANY
Asterisk (*)	Formatted string	ANY
Embedded Question Mark (?)	Formatted string/WFN	Single character ANY
Embedded asterisk (*)	Formatted string/WFN	Multi-character ANY
Unspecified	URI	ANY

DRAFT

5.3 Logical Definitions

This section defines the logical notation that designates the possible outcomes of both CPE attribute comparison and CPE Name matching.

5.3.1 CPE Attribute Comparison Constructs

We apply set theory principles to identify five possible outcomes of a comparison between a source attribute-value pair and a target attribute-value pair. We use common set notation to denote these outcomes. In the context of this specification the sets are determined as subsets of the set of all possible matching values of an attribute. Table 5-2 describes the set notation for these outcomes along with a definition and example for each set relation in the context of CPE attribute comparison. The letters A and B in the Notation column represent source and target values for the same attribute. A version attribute for example could have source value (A) of "3.0", and target value (B) of ANY.

Table 5-2: Outcomes of CPE Attribute Comparison

Notation	Definition	Example
$A \supset B$	The set of possible source attribute values is a SUPERSET of the set of possible attribute values for the target	source = ANY, target = string
$A \subset B$	The set of possible source attribute values is a SUBSET of the set of possible attribute values for the target	source = string, target = ANY
$A = B$	The set of possible attribute values for the source and target are EQUAL	source = NA, target = NA
$A \neq B$	The sets of possible attribute values of the source and target are mutually exclusive or DISJOINT	source=NA, target = string
$A \cap B$	The set of possible attribute values of the source and target INTERSECT	source = partial string match + wild cards, target = partial string match + wild cards

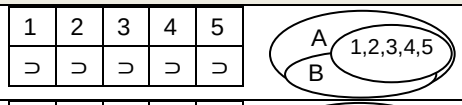
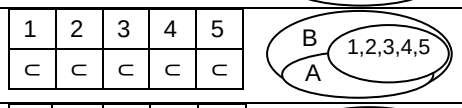
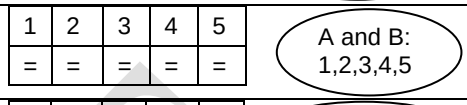
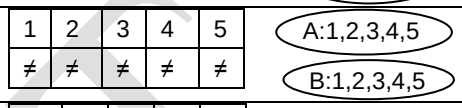
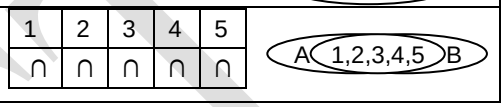
5.3.2 Name Comparison Constructs

This specification defines five possible outcomes of a name level comparison between the set of outcomes resulting from the attribute comparison of a source to a target CPE Name. Attribute comparison outcomes are compared across the set to determine the compound set relation of the source name to the target name. We use common set notation to denote these outcomes.

Table 5-3 describes the set notation for these outcomes along with a definition and example for each set relation in the context of CPE name matching. The letters A and B in the Notation column represent the source and target CPE names respectively.

DRAFT

Table 5-3: Outcomes of CPE Name Comparison

Notation	Definition	Example
A ⊃ B	The set of attribute comparison outcomes for the source name is a SUPERSET of the set of attribute comparison outcomes for the target name	
A ⊂ B	The set of attribute comparison outcomes for the source name is a SUBSET of the set of attribute comparison outcomes for the target name	
A = B	The set of attribute comparison outcomes for the source name is EQUAL to the set of attribute comparison outcomes for the target name	
A ≠ B	The set of attribute comparison outcomes for the source name is DISJOINT with the set of attribute comparison outcomes for the target name	
A ∩ B	The set of attribute comparison outcomes for the source name is an INTERSECT of the set of attribute comparison outcomes for the target name	

5.4 Conditions

This section describes the required preconditions and post conditions of the CPE Name matching process.

5.4.1 Preconditions

1. For purposes of this specification it is assumed that all CPE Names are expressed as well formed names (WFNs). WFNs are unordered sets of CPE attribute-value pairs.
2. Attribute comparison MUST be performed prior to name matching
3. The collective outcome of an attribute comparison of a source CPE Name to a target CPE Name SHALL be used as input to name-level matching

5.4.2 Post Conditions

The CPE Name matching process SHALL provide matching results for each attribute comparison in a CPE Name as well as an overall name match result that reflects one of the set relations that are defined in Table 5-3.

DRAFT

6. Name Matching Methods and Criteria

This section applies set relations to WFN attribute values as defined in Section 5 of this specification (cf. 5.1, 5.3). Section 6.1 enumerates all possible combinations of CPE WFN attribute values and their outcomes. Section 6.2 defines how sets of attribute comparison outcomes are combined to determine the minimal required outcomes of a CPE Name match. Section 6.3 defines matching criteria and methods for matching embedded wild cards in CPE attribute value strings.

6.1 Attribute Comparison

Table 6-1 enumerates all possible combinations of CPE WFN attribute values and their outcomes. The following key describes the attribute value notation for Table 6-1.

1. Column names
    a. Source = The source WFN
    b. Target = The target WFN
    c. Outcome = The required outcome for each source attribute-value to target attribute-value as defined in Section 5.3 of this specification.
2. Cell values
    a. ANY and NA = Logical Values as defined in Section 5.1.2 of this specification.
    b. i = an attribute value string
    c. k = an attribute value string that is not identical to i
    d. string + wild cards = attribute value string with any combination of ? or * embedded wildcards at the beginning or the end of the string.

Table 6-1: Attribute Comparison

Row No.	Source	Target	Outcome
1	ANY	ANY	=
2	ANY	NA	⊃
4	ANY	i	⊃
5	ANY	i + wild cards	⊃
6	NA	ANY	⊂
7	NA	NA	=
9	NA	i	≠
10	NA	i + wild cards	≠
16	i	i	=
17	i	k	≠
18	i	i + wild cards	⊂
19	i	NA	≠
21	i	ANY	⊂
22	i + wild cards	i	⊃
23	i + wild cards	ANY	⊂
24	i + wild cards	k	≠
25	i + wild cards	NA	≠
27	i + wild cards	i + wild cards	=|⊂|⊃†
28	i + wild cards	k + wild cards	≠

†Outcome depends on the string wild card matching algorithm

DRAFT

Wild card usage is a new optional feature in CPE 2.3. See Section 6.3 of this document for a detailed description of wild card matching criteria and methods.

6.2 Name Matching

CPE Name matching is determined by comparing the combined set of attribute outcome results from the attribute comparison phase of the matching process. The results of a CPE Name match are one of five possible CPE Name set relations as defined in Section 5.3.2 of this document.

One of the primary goals of this specification is to provide a foundation that is extensible enough for higher levels in the CPE Specification stack to build upon, while specifying enough commonality to ensure basic interoperability between CPE Name Matching conformant tools. Therefore, the bulk of the baseline commonality of this specification comes from the standardized set relations and the attribute comparison method. The five possible set relation outcomes of attribute comparison and name-level matching will always mean the same thing no matter how we choose to manipulate those outcomes for specialized purposes. This approach allows many degrees of freedom at the name matching level. For this reason, we provide a very small subset of five required name matches in CPE 2.3. We expect that the security automation community will extend CPE Name matching to provide new and innovative ways to define CPE Name matches in order to satisfy various security automation use cases.

The five required CPE Name matches are described in Table 6-2. They apply the attribute comparison method defined in Section 6.1 and the possible CPE Name matching outcomes that are defined in Section 5.3.2 of this specification. Additional name matching outcomes may be identified at higher levels of the CPE specification stack or by the security automation community to meet their operational needs.

Table 6-2: CPE Name Matching Criteria

Name Match Number	If Attribute Outcome	Then Name Match Relation
1	If all attribute outcomes are DISJOINT (≠)	Then CPE name relation = DISJOINT(≠)
2	If all attribute outcomes are EQUAL(=)	Then CPE name relation = EQUAL (=)
3	If all attribute outcome is a SUBSET(⊂)	Then CPE name relation = SUBSET(⊂)
4	If all attribute outcome is a SUPERSET(⊃)	Then CPE name relation = SUPERSET (⊃)
5	If all attribute outcome is INTERSECT (∩)	Then CPE name relation = INTERSECT (∩)

These five name matching outcomes are the minimal required set of name matches for baseline interoperability among CPE Name matching tools. Five corresponding CPE_Name_Compare functions are specified in the pseudo-code example in Section 7.2 of this document that define the expected behavior for meeting the CPE name matching criteria that are defined in this section. CPE implementers who wish to emulate the functionality of the CPE 2.2 Matching algorithm should note that name match numbers 1 and 3 in Table 6-2 produce a final result of FALSE in CPE 2.2, while name match numbers 2 and 4 produce a final result of TRUE.

DRAFT

6.3 Wild Card Attribute Matching

The following sections define matching criteria for wild card matching including the case where wild cards exist in both the source attribute value and corresponding target attribute value of a CPE Name.

6.3.1 Wild Card Attribute Matching Criteria

A wild card CPE attribute value consists of an attribute value string with any combination of question mark (?) or asterisk (*) embedded wildcards at the beginning or the end of the string. The following tables break CPE wild card attribute values into three parts: the start, the end, and the string value. The CPE Name Matching specification does not specify a string comparison method. It only specifies criteria and methods for matching the start and end parts of a wild card attribute value. A start or end could be a wild card character, in which case it will have a character value of either question mark (?) or an asterisk (*). A start or end could also designate the absence of a wild card, which indicates that no wild card exists. Table 6.3.1 below enumerates all possible combinations of start and end values and their outcomes. The following key describes the notation for Table 6-3.

1. Column names
    a. Source = The start or end of a source attribute value
    b. Target = The start or end of a target attribute value
    c. Outcome = The outcome for each corresponding start or end part of an attribute value
2. Cell values
    a. Asterisk (*) wild card as defined in Section 5.1.3 of this specification
    b. Question mark (?) wild card as defined in Section 5.1.3 of this specification
    c. ø = the absence of a wildcard character in a start or end position an attribute value. Please note that the ø character is used in this section for informational purposes only. It does not appear as an actual CPE name character.
    d. ⊃ = a SUPERSET relation as defined in Section 5.3.1
    e. ⊂ = a SUBSET relation as defined in Section 5.3.1
    f. = = an EQUAL relation as defined in Section 5.3.1
    g. ≠ = a DISJOINT relation as defined in Section 5.3.1

Table 6-3: CPE Wild Card Matching Criteria[11]

Source Attribute	Target Attribute	Result
*	*	=
*	?	⊃
*	ø	⊃
?	*	⊂
?	?	=
?	ø	⊃
ø	*	⊂
ø	?	⊂
ø	ø	≠

[11] The set relations in this context refer only to the start and end part of an attribute value. They do not apply to string matching. For example, a ≠ indicates a DISJOINT wild card comparison meaning there is no overlap between the set of possible source results and the set of possible target results.

DRAFT

6.3.2 Wild Card Attribute Matching Methods

This section specifies the conditions and outcomes for wild card CPE attribute value matching. It assumes that a function (Fn) exists which scans a source string for all occurrences of a target string and returns the starting index of each occurrence. Each of the following matching tables defines wild card matching criteria under the conditions of one possible outcomes of this scan.

1. Table 6-4 addresses the condition where source string and target string are EQUAL
2. Table 6-5 addresses the condition where source string is a SUPERSET of target string
3. Table 6-6 addresses the condition where source string is a SUBSET of target string

Table 6-4: Where Source and Target Strings are EQUAL

Source	Target	Result
⊃	⊃	⊃
⊃	⊂	⋂
⊃	=	⊃
⊃	≠	⊃
⊂	⊃	⋂
⊂	⊂	⊂
⊂	=	⊂
⊂	≠	⊂
=	⊃	⊃
=	⊂	⊂
=	=	=
=	≠	=
≠	⊃	⊃
≠	⊂	⊂
≠	=	=
≠	≠	=

Table 6-5: Where Source String is a SUPERSET of Target String

Source	Target	If Condition	Then Result
*	*	in all cases	⊂
*	?	iff Fn has a result equal to length(source) - length(target) - 1	⊂
*	ø	iff Fn has a result equal to length(source) - length(target)	⊂
?	*	iff Fn has a result equal to 2	⊂
?	?	iff Fn has a result equal to 2 and length(source) - 2 = length(target)	⊂
?	ø	iff Fn has a result equal to 2 and length(source) - 1 = length(target)	⊂
ø	*	iff Fn has a result equal to 1	⊂
ø	?	iff Fn has a result equal to 1 and length(source) - 1 = length(target)	⊂
ø	ø	in all cases	≠

Table 6-6: Where Source string is a SUBSET of target string

Source	Target	If Condition	Then Result
*	*	in all cases	⊂
*	?	iff Fn has a result equal to length(source) - length(target) - 1	⊂
*	ø	iff Fn has a result equal to length(source) - length(target)	⊂
?	*	iff Fn has a result equal to 2	⊂
?	?	iff Fn has a result equal to 2 and length(source) - 2 = length(target)	⊂
?	ø	iff Fn has a result equal to 2 and length(source) - 1 = length(target)	⊂
Ø	*	iff Fn has a result equal to 1	⊂
Ø	?	iff Fn has a result equal to 1 and length(source) - 1 = length(target)	⊂
Ø	ø	in all cases	≠

DRAFT

7. CPE Name Matching Pseudo-Code

This section specifies the required common matching capability in terms of an abstract pseudo-code programming language to specify intended computational behavior. Pseudo-code is intended to be straightforwardly readable and translatable into real programming language terms. In reading pseudo-code the following notes should be kept in mind:

- All pseudo-code functions are *pass by reference*, meaning that any changes applied to the supplied arguments within the scope of the function do not affect the values of the variables in the caller's scope.
- In a few cases, the pseudo-code functions reference (more or less) standard library functions, particularly to support string handling. Whenever possible, we reference semantically equivalent functions from the GNU C library, (cf. http://www.gnu.org/software/libc/manual/html_node/index.html#toc_String-and-Array-Utilities).

7.1 Summary of CPE Name Matching Pseudo-code

Table 7-1 provides a line-by-line summary of the pseudo-code in Section 7.2.

Table 7-1: Description of the CPE Name Matching Algorithm

Line Number(s)	Description
1	Begins the attribute comparison function
2	Creates a new associative array table
3 – 13	Compares each source attribute value to its corresponding target value
14	Returns the combined outcome results of the attribute comparison
15	Ends the comparison function that began on Line 1
16	Begins the function definition for the attribute comparison
17 – 19	Defines the attribute outcome as EQUAL (=) if both source and target values are both ANY
20 – 22	Defines the attribute outcome as target is a SUBSET of source (⊃) if source value is ANY
23 – 25	Defines the attribute outcome as source is a SUBSET of target (⊂) if target value is ANY
26 – 28	Defines the attribute outcome as EQUAL (=) if both source and target values are an exact match
29 – 31	Defines the attribute outcome as DISJOINT (≠) if either source or target values are NA
32	Ends the function definition that began on Line 16
33	Begins the first of 5 function definitions for name-level comparison the DISJOINT function;
34-40	Defines the DISJOINT (≠) match as TRUE if all attribute outcomes are DISJOINT (≠)
41	Defines the DISJOINT match as otherwise FALSE
42	Ends the DISJOINT function definition that began on Line 33
43	Begins the SUBSET function;
44-50	Defines the SUBSET (⊂) match as TRUE if all attribute outcomes are SUBSET (⊂)
51	Defines the SUBSET match as otherwise FALSE

DRAFT

Line Number(s)	Description
52	Ends the SUBSET function definition that began on Line 43
53	Begins the SUPERSET function;
54-60	Defines the SUPERSET (⊃) match as TRUE if all attribute outcomes are SUPERSET (⊃)
61	Defines the SUPERSET match as otherwise FALSE
62	Ends the SUPERSET function definition that began on Line 53
63	Begins the EQUAL function;
64-70	Defines the EQUAL (=) match as TRUE if all attribute outcomes are EQUAL (=)
71	Defines the EQUAL match as otherwise FALSE
72	Ends the EQUAL function definition that began on Line 63
73	Begins the INTERSECT function;
74-80	Defines the INTERSECT (∩) match as TRUE if all attribute outcomes are INTERSECT (∩)
81	Defines the INTERSECT (∩) match as otherwise FALSE
82	Ends the INTERSECT (∩) function definition that began on Line 73

7.2 CPE Name Match Pseudo-code

The following CPE Name Matching pseudo-code defines the functions that are described in Section 7.1 of this document.

```
1 function CPE_Attribute_Compare(source, target)
    ;; Compare each attribute of the Source WFN to the Target WFN.
2   result := new table.
    ;; compare results from the get function defined in Section 5.6.2
    ;; of the CPE Naming Specification.
3   put(result, part, compare(get(source, part), get(target, part))).
4   put(result, vendor, compare(get(source, vendor), get(target,
    vendor))).
5   put(result, product, compare(get(source, product), get(target,
    product))).
6   put(result, version, compare(get(source, version), get(target,
    version))).
7   put(result, update, compare compare(get(source, update),
    get(target, update))).
8   put(result, edition, compare(get(source, edition), get(target,
    edition))).
9   put(result, language, compare(get(source, language), get(target,
    language))).
10  put(result, sw_edition, compare(get(source, sw_edition),
    get(target, sw_edition))).
11  put(result, target_sw, compare(get(source, target_sw),
    get(target, target_sw))).
12  put(result, target_hw, compare(get(source, target_hw),
    get(target, target_hw))).
```

```
13  put(result, other, compare(get(source, other), get(target,
    other))).
    ;; For each attribute comparison, return one of the attribute
    ;; outcome results as defined in the attribute comparison table
    ;; in Section 6.1 of this document.
14   return result.
15 end
   ;; Defines the Attribute_Compare function.
   ;; The result is the set of attribute outcomes as defined in
   ;; Section 6.2 of this document.
16 function compare(source, target)
17  if (source = ANY and target = ANY) then
18     return =.
19  end
    ;; If both source and target attribute values are ANY, then the
    ;; attributeOutcome as EQUAL (=).
20  if (source = ANY) then
21     return ⊃.
22  end
    ;; If source attribute value is ANY, then the attributeOutcome is
    ;; a SUPERSET.
23  if (target = ANY) then
24     return ⊂.
25  end
    ;; If Target attribute value is ANY, then the attributeOutcome is
    ;; a SUBSET.
26  if (target = source) then
27     return =.
28  end
    ;; If source and target values are an exact match then the
    ;; attributeOutcome is EQUAL (=).
29  if (target = NA or source = NA) then
30     return ≠.
31  end
    ;; If either source or target attribute value is NA then the
    ;; attributeOutcome is DISJOINT (≠).
32  end

    ;; CPE wild card matching criteria and behavior are defined
    ;; in Section 6.3 of this document
    ;; For the sake of brevity, no pseudo-code example is provided.

    ;; Begin five examples CPE name matching functions as defined by
    ;; the criteria in Section 6.2 of this document.

   ;; Begin CPE DISJOINT function.
33 function CPE_DISJOINT(source, target)
34 result := CPE_Name_Compare(source, target).
    ;; Compare combined sets of attribute outcomes to match source
    ;; to target WFNs. If all attribute outcomes are DISJOINT (≠) then
    ;; CPE name relationship is DISJOINT(≠).
```

```
35  if (attributeOutcome(result, =) = false and
36      attributeOutcome(result, ⊃) = false and
37      attributeOutcome(result, ⊂) = false and
38      attributeOutcome(result, ∩) = false) then
39     return TRUE.
40  end
41  return FALSE.
42 end ;; Ends CPE DISJOINT function.

   ;; Begin CPE SUBSET function.
43 function CPE_SUBSET(source, target)
44 result := CPE_Name_Compare(source, target).
    ;; Compare combined sets of attribute outcomes to match source
    ;; to target WFNs. If all attribute outcomes are SUBSET (⊂) then
    ;; CPE name relationship is SUBSET (⊂).
45  if (attributeOutcome(result, ≠) = false and
46      attributeOutcome(result, =) = false and
47      attributeOutcome(result, ⊃) = false and
48      attributeOutcome(result, ∩) = false) then
49 return TRUE.
50  end
51  return FALSE.
52 end ;; Ends CPE_SUBSET function.

   ;; Begin CPE SUPERSET function.
53 function CPE_SUPERSET(source, target)
54 result := CPE_Name_Compare(source, target).
    ;; Compare combined sets of attribute outcomes to match source
    ;; to target WFNs. If all attribute outcomes are SUPERSET (⊃) then
    ;; CPE name relationship is SUPERSET (⊃).
55  if (attributeOutcome(result, ≠) = false and
56      attributeOutcome(result, =) = false and
57      attributeOutcome(result,⊂) = false and
58      attributeOutcome(result, ∩) = false) then
59 return TRUE.
60  end
61  return FALSE.
62 end ;; Ends CPE SUPERSET function.

   ;; Begin CPE EQUAL function.
63 function CPE_EQUAL(source, target)
64 result := CPE_Name_Compare(source, target).
    ;; Compare combined sets of attribute outcomes to match source
    ;; to target WFNs. If all attribute outcomes are EQUAL (=) then
    ;; CPE name relationship is EQUAL (=).
65  if (attributeOutcome(result, ≠) = false and
66      attributeOutcome(result, ⊃) = false and
67      attributeOutcome(result,⊂) = false and
68      attributeOutcome(result, ∩) = false) then
69 return TRUE.
```

```
70  end
71  return FALSE.
72 end ;; Ends CPE EQUAL function.

   ;; Begin CPE INTERSECT function.
73 function CPE_INTERSECT(source, target)
74 result := CPE_Name_Compare(source, target).
    ;; Compare combined sets of attribute outcomes to match source
    ;; to target WFNs. If all attribute outcomes INTERSECT (∩) then
    ;; CPE name relationship is INTERSECT (∩).
75  if (attributeOutcome(result, ≠) = false and
76      attributeOutcome(result, ⊃) = false and
77      attributeOutcome(result, ⊂) = false and
78      attributeOutcome(result, =) = false) then
79 return TRUE.
80  end
81  return FALSE.
82 end ;; Ends CPE INTERSECT function.
```

DRAFT

Appendix A—Change Log

Release 0 – 9 June 2010

- Initial draft specification released to the CPE community as a read ahead for the CPE Developer Days Workshop

Release 1 – 30 June 2010

- Near final draft released to NIST for submission to review process
- Minor editorial changes throughout the document
- Added abstract and change log sections
- Removed all mention of and support for the logical value UNKNOWN
- Updated audience sections to align with the CPE Naming specification
- Updated the name matching sections to reflect the new intersection relation, the set relation matching results and the minimal required name matching criteria
- Added restrictions to the wild card verbiage to allow only start and end wild card usage within a value string
- Added source wild card to target wild card matching pseudo-code
- Broke out the name matching function to four separate functions and added a function for the new intersection relation to the pseudo-code
- Added a new section to define wild card matching criteria and methods

DRAFT